PERGAMON

Neural Networks 14 (2001) 1201–1218

Neural Networks

www.elsevier.com/locate/neunet

Contributed article

A closed-form neural network for discriminatory feature extraction from high-dimensional data

Ashit Talukder[a], David Casasent[b,*]

[a]*Jet Propulsion Laboratory/California Institute of Technology, Pasadena, CA 91109, USA*
[b]*Department of Electrical and Computer Engineering, Carnegie Mellon University, Pittsburgh, PA 15213, USA*

Abstract

We consider a new neural network for data discrimination in pattern recognition applications. We refer to this as a maximum discriminating feature (MDF) neural network. Its weights are obtained in closed-form, thereby overcoming problems associated with other nonlinear neural networks. It uses neuron activation functions that are dynamically chosen based on the application. It is theoretically shown to provide nonlinear transforms of the input data that are more general than those provided by other nonlinear multilayer perceptron neural network and support-vector machine techniques for cases involving high-dimensional (image) inputs where training data are limited *and* the classes are not linearly separable. We experimentally verify this on synthetic examples. © 2001 Elsevier Science Ltd. All rights reserved.

Keywords: Dimensionality reduction; Discrimination; Feature extraction (nonlinear); Neural networks; Nonlinear transform; Pattern recognition

1. Introduction

In this paper, we consider a new nonlinear neural network-based feature extraction procedure for pattern recognition applications. It is referred to as the maximum discriminating feature (MDF) neural network, since it is capable of extracting features that are useful for discrimination.

The Fisher linear discriminant (Fisher, 1950) is a linear transformation that is well suited for separating image/signal data for different objects or classes. It yields fewer discriminant feature vectors than the number of classes, i.e. only $c-1$ feature vectors when c classes are present. This is restrictive in many pattern recognition problems where more than c features are necessary. The Fisher linear discriminant and its variations such as the orthonormal discriminant vector (ODV) (Foley & Sammon, 1975; Hamamoto, Kanaoka & Tomita, 1993) consider the squared separation between the *means* of each class and, hence, cannot perform well on data with multiple clusters per class. Our MDF neural network feature extraction approach uses a new discrimination measure that handles data with multiple clusters per class.

The main advantage of linear transforms is that they are easy to design and typically have closed-form solutions. However, linear transforms typically extract information from only the second-order correlations in the data (covariance matrix) and ignore higher-order correlations in the data. It has been shown that many signals in the real world are inherently non-symmetric (Barlow, 1961; Softky & Kammen, 1991) and, therefore, contain higher-order correlation information that could be useful; linear principal component analysis (PCA) is incapable of representing such data (Softky & Kammen, 1991). For such cases, nonlinear transforms are necessary. A number of nonlinear transformation methods for pattern recognition exist (Cortes & Vapnik, 1995; Karhunen & Joutsensalo, 1994; Kramer, 1991; Lippmann, 1987; Scholkopf, Smola & Muller, 1998). Neural networks are among the most commonly used nonlinear techniques, since they provide very general transforms (Lippmann, 1987). It has been proven that artificial neural networks can approximate posterior class probabilities (Ruck, Rogers, Kabrisky, Oxley & Suter, 1990) and are, therefore, useful for class discrimination (classification). It has been shown (Bishop, 1995) that neural networks with threshold activation functions can produce arbitrary piecewise linear decision surfaces; the number of hidden layer neurons required increases significantly with the complexity of the decision surface, which results in larger training set requirements (Baum & Haussler, 1989). Other neural networks inherently produce piecewise-hyperquadratic (Telfer & Casasent, 1993) surfaces, and general piecewise quadratic (Casasent & Natarajan, 1995) boundaries and, hence, require fewer hidden-layer neurons to produce piecewise closed regions.

* Corresponding author. Tel.: +1-412-268-2464; fax: +1-412-268-6345.
E-mail address: casasent@ece.cmu.edu (D. Casasent).

Neural networks typically have a number of ad hoc parameters (Casasent, Neiberg & Sipe, 1998), such as selection of the number of hidden layers, the number of hidden-layer neurons, parameters associated with the learning or optimization technique used, and in many cases require a validation set for the stopping criterion. In addition, the weights in a neural network are trained iteratively and this produces problems with convergence to local minima. Our nonlinear MDF neural network weights are obtained in closed-form and, thus, do not have problems associated with iterative neural network solutions.

A number of neural networks, termed nonlinear PCA neural networks (Baldi & Hornik, 1989; Karhunen & Joutsensalo, 1994, 1995; Kramer, 1991) attempt to perform dimensionality reduction of the input data either using nonlinear transforms or by considering higher-order correlations in the input data. These prior iterative nonlinear PCA neural network techniques to select weights are noted to have convergence problems (Karhunen & Joutsensalo, 1995), in contrast to our linear and nonlinear MDF algorithms that have closed-form solutions. We discuss nonlinear PCA neural networks in more detail and discuss the types of nonlinear transforms they can produce in Section 4.

We now discuss prior work on nonlinear transformation procedures applied to high-dimensional (image) input data, including kernel-based methods (Cortes & Vapnik, 1995; Scholkopf et al., 1998). All of these methods consider higher-order cross-terms in the input data to produce nonlinear transforms of the input data. This approach of augmenting input data with cross-product terms to solve pattern recognition problems has been noted elsewhere (Cover, 1965; Duda & Hart, 1973, pp. 135–137; Gheen, 1994). The concept of nonlinearly transforming each input pattern has been noted to theoretically yield better class separability (Cover, 1965). However, due to the "curse of dimensionality", it has been noted that such solutions are typically not easy to obtain due to the presence of limited training data. Recently a nonlinear feature extraction kernel PCA procedure with a closed-form solution has been formulated (Scholkopf et al., 1998). However, kernel PCA is designed merely for data representation, not for discrimination; our MDF algorithm is useful for data discrimination. Volterra filters (Gheen, 1994) require iterative solutions; our solution is a closed-form one. Another nonlinear *classification* technique, the support vector machine (SVM) (Boser, Guyon & Vapnik, 1992; Cortes & Vapnik, 1995) uses only a few training set samples referred to as support vectors, those that lie close to samples in another class. The polynomial SVM technique has associated problems in terms of the generality of the transformations produced and the number of on-line computations (as detailed in Section 4).

Our MDF neural network uses new dimensionality reduction techniques for classification of high-dimensional input data in the presence of small training sets. Its weights are obtained in closed-form, and its activation functions are dynamically chosen, thereby increasing the types of nonlinear transforms it can produce. Our emphasis in this paper is on applications involving high-dimensional input data where sufficient training data are unavailable. We compare the performance of different pattern recognition techniques on such high-dimensional data, both theoretically and through experiments. The experimental data that we use are synthetically generated with well-known probability distributions (and, therefore, with well known decision surfaces), rather than images with unknown distributions; this allows us to compare the decision surfaces obtained with the different pattern recognition methods with that of the ideal decision surface.

In Section 2, we detail our nonlinear MDF neural network technique. Our new solution for high-dimensional input data (Section 3) is then derived. High-dimensional input data correspond to cases where the total number of training samples S is less than the dimensionality N of the input, $S < N$. For image input data, N is the number of pixels and typically, $S \ll N$. We also discuss the types of nonlinear transformations (decision boundaries) obtained for high-dimensional inputs by our nonlinear MDFs in Section 3. We theoretically compare these results with those for other nonlinear transformation methods on high-dimensional input data only, including kernel-based approaches such as kernel PCA, SVMs, and nonlinear PCA neural networks for dimensionality reduction in Section 4. The on-line computation load for the different methods are also noted. We also show in Section 4 that our MDF algorithm allows transforms of any polynomial order and that the order of the transform and the parameters in the MDF algorithm are automatically selected. This is not the case with polynomial kernel-based methods such as PCA and SVMs. *Our theoretical analysis of the nonlinear transformations produced by nonlinear kernel-based methods and nonlinear PCA neural networks is unique and, to our knowledge, has not been discussed in any prior work.* Our theoretical analyses are verified through experiments on synthetic data in Section 5.

We first discuss some of the terminology we use for nonlinear transformations. The parameters we use to describe a given nonlinear transform are its rank, principal orientation, and degree. The number of dimensions in which a nonlinear transform curves in the input space is referred to as the *rank* of the transform. This is an extension of the well-known concept of the rank of quadratic transforms (Weber & Casasent, 1998). Note that when closed decision surfaces are necessary in an N-dimension input space, the rank of the nonlinear transform should be N. The *degree* of a nonlinear transform is the number of non-zero derivatives that the function describing it has. The degree is also the polynomial order with which functions can be approximated using a Taylor series expansion (Spiegel, 1983). Another issue is the directions in which the nonlinear transform can be oriented in the input space. We refer to the orientation or direction along which the maximum curvature occurs as the

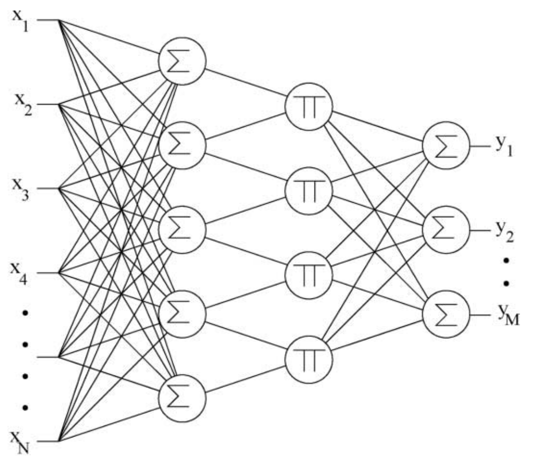

Fig. 1. Neural network implementation of the quadratic nonlinear MDF for low input dimensions.

principal orientation of the transform. This is an extension of the concept for elliptical or hyperelliptical quadratic transforms where the major axis of the ellipse is the direction along which the maximum curvature in the output quadratic transform occurs. We feel that transforms with a *larger rank* and *more possible principal orientations* are preferable since they are more general. We also feel that nonlinear transforms with adjustable (flexible) degrees are more powerful than ones with a fixed degree. We verify this through experiments. In Sections 3 and 4, we will address the above three properties for our nonlinear MDF algorithm and for other nonlinear transformation methods on high-dimensional input data where $S < N$ or $S \ll N$.

2. Nonlinear MDF formulation: measure and general solution

In this section, we detail our nonlinear MDF neural network concept. We first define the discrimination measure we use in the MDF neural network for pattern recognition problems involving discrimination or classification. The weights of the MDF neural network are the solutions that maximize this discrimination measure. For simplicity of explanation, we refer to the MDF neural network weights as transformation vectors.

Deterministic vectors and matrices are represented by lower and upper-case letters with underlines ($\underline{x}$ and $\underline{X}$). Random variables are lower-case letters in sans-serif font (x), and random vectors are lower-case bold ($\mathbf{x}$). Classes are represented as random vectors $\mathbf{x}_1$, $\mathbf{x}_2$ with corresponding mean vectors $\underline{\mu}_1$, $\underline{\mu}_2$ and covariance matrices $\underline{C}_1$, $\underline{C}_2$. The mean μ and the covariance $\underline{C}$ of a random vector $\mathbf{x}$ are $\underline{\mu} = E[\mathbf{x}]$ and $\underline{C} = E[\mathbf{x}\mathbf{x}^T] - \underline{\mu}\,\underline{\mu}^T$. These statistics are typically estimated from samples and, hence, only sample (estimated) means and covariance matrices are available. When considering a set of S sample data vectors in class 1, $\{\underline{x}_{1s}\}$, and Q samples in class 2, $\{\underline{x}_{2q}\}$, we describe them by the sample data matrices $\underline{X}_1 = [\underline{x}_{11}\ \underline{x}_{12}\ \cdots\ \underline{x}_{1S}]$ and $\underline{X}_2 = [\underline{x}_{21}\ \underline{x}_{22}\ \cdots\ \underline{x}_{2Q}]$.

Our objective is to find a set of nonlinear transformations (MDF neural network weights) on $\mathbf{x}$ that optimally discriminate between multiple classes in a reduced dimensionality space. A general nonlinear transform on a random vector $\mathbf{x}$ is $\underline{y} = f(\mathbf{x})$ and yields a random variable (feature) y. We use techniques by which we express the nonlinear transformation as a linear one; this allows us to directly use earlier linear techniques developed to obtain closed-form solutions for the nonlinear transformations. We first nonlinearly transform or map the input data into another space, and then apply linear solutions for the MDF weights in this space. Therefore, given an input vector $\underline{x}$, we transform it nonlinearly into a nonlinear data space, $\mathbf{x}_H = \Psi(\mathbf{x})$. For example, the nonlinear mapping $\Psi(\,)$ could be a quadratic polynomial transformation that contains linear and second-order input cross terms. Then, the data mapped to the nonlinear space would be

$$\mathbf{x}_H = [\,x_1 x_2 \quad \cdots \quad x_N \quad x_1 x_1 \quad x_1 x_2 \quad \cdots \quad x_N x_N\,]^T, \qquad (1)$$

where $H = N + N(N-1)/2$ is the dimension of $\mathbf{x}_H$. $\mathbf{x}_H$ contains higher-order terms in the original input random vector $\mathbf{x}$. This higher-dimensional vector has N linear terms and $N(N-1)/2$ unique nonlinear cross-product terms. This higher-order and higher-dimensional data vector is *linearly* transformed to yield an output feature (random variable) $\mathsf{y} = \underline{\phi}^T \mathbf{x}_H$. Since the quadratic transform is written as a linear transform with combination coefficients (weights) $\{\phi_i\}$, linear solution techniques are used to obtain the values for the coefficients $\{\phi_i\}$ for a given application. Note that a quadratic transform can also be written in matrix form as $\mathsf{y} = \underline{b}^T\mathbf{x} + \mathbf{x}^T\underline{A}\mathbf{x}$ where $\underline{A}$ is a matrix and $\underline{b}$ is a vector. We use the formulation in Eq. (1), since it allows us to write the quadratic transform as a linear one and, therefore, *allows for a closed form solution in the* $\mathbf{x}_H$ *space that is quadratic in the original* $\mathbf{x}$ *space*. Note that this can be generalized easily to yield polynomial mappings, of any arbitrary order. The mappings to obtain $\mathbf{x}_H$ are not necessarily restricted to polynomial transforms.

Note that when the mapping $\mathbf{x}_H$ is a quadratic transform, it can be expressed as a neural network with summation and product neurons. Fig. 1 shows a neural network implementation of the quadratic MDF with N input dimensions and M outputs (features). The neural network implementation of the quadratic MDF is exactly the same as a Sigma–Pi neural network. The Sigma–Pi neural network (Heywood & Noakes, 1995) has linear summation (Σ) and nonlinear product (Π) neurons; the number of inputs to the Π neurons indicates the polynomial order of the neural network.

To derive our second-order nonlinear MDF algorithm, we consider two classes of data described by random vectors $\mathbf{x}_1$ and $\mathbf{x}_2$. We create the augmented random vectors, $\mathbf{x}_{1H}$ and $\mathbf{x}_{2H}$ for each class; these contain first- and second-order terms as in Eq. (1). The corresponding augmented data

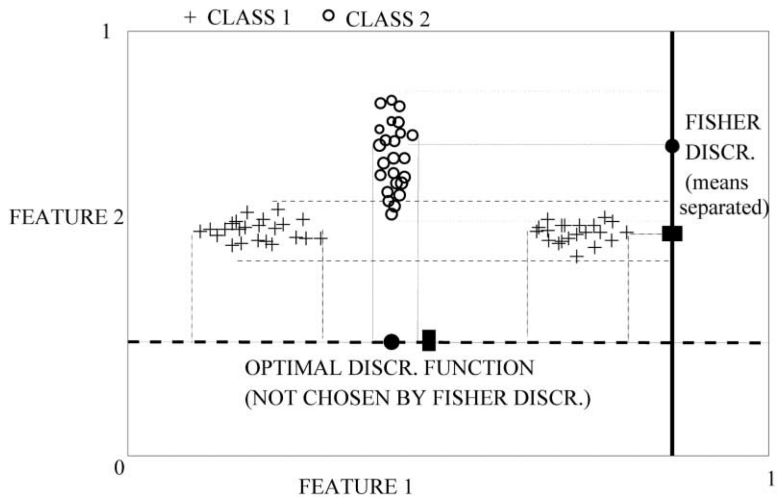

Fig. 2. Example when the Fisher linear discriminant yields bad discrimination.

vectors for the two classes are $\{\underline{x}_{1sH}\}$ and $\{\underline{x}_{2H}\}$. We desire to determine the M best transforms $\underline{\Phi}_M = [\underline{\phi}_1\ \underline{\phi}_2\ \cdots\ \underline{\phi}_M]$ such that the transformed random vectors (output features) for the two classes $\mathbf{y}_1 = \underline{\Phi}_M^T\mathbf{x}_{1H}$ and $\mathbf{y}_2 = \underline{\Phi}_M^T\mathbf{x}_{2H}$ are separated. Here, $\underline{\Phi}_M = [\underline{\phi}_1\ \underline{\phi}_2\ \cdots\ \underline{\phi}_M]$ is an $H \times M$ matrix containing M basis function vectors $\underline{\phi}_m$. Each output feature (random variable) y_m of the feature vector (random vector) $\mathbf{y}$ is the nonlinear (quadratic) transformation of the input random vector $\mathbf{x}$ using a transformation vector $\underline{\phi}_m$. This is the general notation we use in this paper.

2.1. MDF discrimination measure

For discrimination, we want the projections (transformed values) for each class to be separated in our new feature space. Measures such as the square of the difference in the projections of the class means that is used in the Fisher and the ODV techniques are not robust enough to handle many class distributions. Fig. 2 shows an example of a 2-D input discrimination problem in which one class (‘O’) has one cluster and the second class (‘ + ’) has two clusters. The Fisher linear discriminant (the vertical line in Fig. 2) separates the *means* of the two classes and, thus, has some overlap between the two classes. The optimal linear discrimination in this case is a horizontal line. The sample projections of both classes on it are well separated, even though the means of the two class projections are not. We, thus, use the average squared difference of *all* projection values for two classes. *We expect this to be a better measure of separation, since it does not use the means of the class projections*. As before, we initially consider two classes.

For discrimination, our goal is to compute M nonlinear transformation vectors $\underline{\phi}_m$, $m = 1, 2, \ldots, M$ such that the output features (random variables) $y_{1m} = \underline{\phi}_m^T\mathbf{x}_{1H}$ for class 1, and $y_{2m} = \underline{\phi}_m^T\mathbf{x}_{2H}$ for class 2 are well separated from each other (in the statistical sense). Therefore, the $\underline{\phi}_m$ should maximize $E[(y_{1m} - y_{2m})(y_{1m} - y_{2m})^T]$; or equivalently $\underline{\phi}_m$ is chosen to maximize $E[\underline{\phi}_m^T(\mathbf{x}_{1H} - \mathbf{x}_{2H})(\mathbf{x}_{1H} - \mathbf{x}_{2H})^T\underline{\phi}_m]$. This can be written in more compact form as maximizing $E[\phi_m^T(x_{1H}\underline{R}_{12}\underline{\phi}_m]$, where $\underline{R}_{12} = E[(\mathbf{x}_{1H} - \mathbf{x}_{2H})(\mathbf{x}_{1H} - \mathbf{x}_{2H})^T]$ is a vector-outer-product difference matrix. To better understand the intuitive reasoning behind use of this discrimination measure, we consider sampled data. We denote the output of a class 1 sample using the nonlinear transformation vector $\underline{\phi}_m$ by $y_{\mathrm{m1s}} = \underline{\phi}_{\mathrm{m}}^{\mathrm{T}}\underline{x}_{1\mathrm{sH}}$; similarly, the output of a class 2 sample is $y_{\mathrm{m2q}} = \underline{\phi}_{\mathrm{m}}^{\mathrm{T}}\underline{x}_{2qH}$. For the projections of the two classes to be best separated, we desire that $[(y_{\mathrm{m1_s}} - \mathrm{y}_{\mathrm{m2_q}})^2]$ be large (or maximized) for all samples s and q and for all nonlinear transformation vectors $\underline{\phi}_m$, $1 \leq m \leq M$, i.e. we maximize the mean squared separation between *all* class 1 and class 2 outputs using each transformation vector $\underline{\phi}_m$. This measure does not separate the means of the two classes; instead it separates *all* pairs of samples in the two classes. Thus, it can inherently handle classes with multiple clusters. This property is also desirable for problems involving $L > 2$ classes where, implicitly, at least L clusters are present (i.e. at least one cluster per class). Measures such as the Fisher linear discriminant and the ODV cannot handle multiple clusters and are, therefore, not well suited for $L > 2$ class discrimination problems. For the discrimination measure in the MDF algorithm, we also minimize the spread of the transformed samples of each class, i.e. we minimize $\underline{\phi}_m^T(\underline{C}_{1H} + \underline{C}_{2H})\underline{\phi}_m$ for all $\underline{\phi}_m$, where $\underline{C}_{1H} = E[\mathbf{x}_{1H}\mathbf{x}_{1H}^T] - \underline{\mu}_{1H}\underline{\mu}_{1H}^T$ is the higher-order covariance matrix of class 1 and $\underline{\mu}_{1H} = E[\mathbf{x}_{1H}]$. Similarly, $\underline{\mathrm{C}}_{2H}$ is the higher-order covariance matrix of class 2. Minimizing the sum of the output covariances for the two classes ensures minimal intra-class spread in the output feature space. Therefore, to determine the best set of transformation coefficients $\underline{\phi}_m$ that separate classes 1 and 2, we maximize the *new discrimination measure*

$$E_{\mathrm{D}} = \sum_{m=1}^{M} E_{\mathrm{D}m} = \sum_{m=1}^{M} \frac{\underline{\phi}_m^T\underline{R}_{12H}\underline{\phi}_m}{\underline{\phi}_m^T(\underline{C}_{1H} + \underline{C}_{2H})\phi_m}, \tag{2}$$

where $\underline{R}_{12H} = E[(\mathbf{x}_{1H} - \mathbf{x}_{2H})(\mathbf{x}_{1H} - \mathbf{x}_{2H})^T]$. An estimate of this $\underline{R}_{12H}$ matrix is computed from the data samples in classes 1 and 2 as $\underline{R}_{12H} = (1/SQ)\sum_{q=1}^{Q}\sum_{s=1}^{S}(\underline{x}_{1sH} - \underline{x}_{2qH})(\underline{x}_{1sH} - \underline{x}_{2qH})^T]$.

2.2. MDF solution

The nonlinear transformation vectors that best separate two classes are those that maximize the discrimination measure in Eq. (2). Differentiating Eq. (2) with respect to $\underline{\Phi}_m$, the $\underline{\phi}_m$ solution must satisfy

$$[\underline{C}_{1H} + \underline{C}_{2H}]^{-1}\underline{R}_{12H}\underline{\Phi}_M = \underline{\Phi}_M\underline{\Lambda}. \quad (3)$$

Thus, the M best nonlinear MDF basis functions $\underline{\phi}_m$ are the M dominant eigenvectors of

$$[\underline{C}_{1H} + \underline{C}_{2H}]^{-1}\underline{R}_{12H}. \quad (4)$$

The MDF algorithm automatically finds a closed-form solution for the best set of given nonlinear transforms. The performance measure in Eq. (2) is a new contribution of this work, as it allows data with multiple clusters per class. The solution in Eq. (4) is only realistic for low-dimensionality input data. Section 3 notes our new solution for the high-dimensionality input data case.

3. Nonlinear MDF neural network for high-dimensional data

For the high-dimensionality input data case of concern, we restrict the nonlinear transforms used in the MDF to polynomial ones. By varying the polynomial order used, the shape of the resultant nonlinear decision boundaries produced changes and allows for very general decision boundaries as we will detail.

For image input data with high dimensionality N, one issue in using general polynomial transformations is that the on-line computation time (calculation of the VIPs $y = \underline{\phi}_m^T\underline{x}_H$) is high $\mathcal{O}(N^p)$, where p is the order of the polynomial transformation ($p = 2$ in the quadratic case). Another issue is that the total number of training samples S is now significantly less than the dimensionality of the input ($S < N$ and typically $S \ll N$); when $S < N$, the sample covariance matrix $\underline{C}_H$ in Section 2 is degenerate and, hence, cannot be inverted as required in Eq. (4). To solve these problems, we use a restricted polynomial transform and a new two-stage algorithm. We first transform the input image to a low-dimensional $M < S$ feature space using a new performance measure that does not involve matrix inversions. In the second stage, we then use the MDF measure in Eq. (2) in this lower (M) dimensional space where the matrix inversion of the covariance matrix is easy since $M < S$. *This entire approach is very new*. We first discuss the general concept of the type of transform we use for image inputs and then detail our new two-stage MDF procedure.

To avoid on-line computation problems, we use a restricted nonlinear polynomial transformation that neglects cross-product input data terms in the first stage. Specifically, we raise the input data $\mathbf{x} = [\mathrm{x}_1 \cdots \mathrm{x}_N]^T$ (corresponding to pixels of the input image, when the input data is an image) to the power p' to produce the associated nonlinear input vector $\mathbf{x}_{p'}$ and new first-stage output features

$$y = \underline{\phi}'^T\mathbf{x}_{p'} = \sum_{i=1}^{N}\phi'_i x_i^{p'}, \text{ where } \mathbf{x}_{p'} = [\, x_1^{p'} \quad x_2^{p'} \quad \cdots \quad x_N^{p'}\,]^T; \quad (5)$$

The computation time for this first-stage operation is linear in N,

$$\mathcal{O}(3N) + \mathcal{O}(N) = \mathcal{O}(4N) \simeq \mathcal{O}(N). \quad (6)$$

In Eq. (6), we assume that raising *each* pixel to a power p' requires three operations (a log operation, followed by a multiplication by the polynomial power p' and an antilog operation), and that the vector inner product (VIP) requires N computations (since it does not involve cross-product terms).

For the first stage, we choose the transform $\underline{\Phi}'_M$ to maximize the *new discrimination measure* $E'_\mathrm{D} = \underline{\Phi}'^T_M\underline{R}_{12p'}\underline{\Phi}'_M$ (i.e. there is no denominator as in Eq. (2)). Here, $\underline{R}_{12p'}$ is calculated for the new data vectors $\underline{x}_{p'}$ and contains higher-order correlation information about the input data. The $\underline{\Phi}'_M$ solutions are the dominant eigenvectors of $\underline{R}_{12p'}$. Note that the new discrimination measure E'_D *does not require a matrix inversion as occurred in* Eq. (2) *and, thus, computation of the transforms* $\underline{\phi}'_M$ *to use is easy and possible*. These $\underline{\phi}'_M$ transform an image input of dimension N into a nonlinear feature space $\mathbf{y}'_M = \underline{\Phi}'^T_M\mathbf{x}_{p'}$ *of reduced dimension* $M \leq S \ll N$. The matrix $\underline{\mathrm{R}}_{12p'}$ is symmetric and, therefore, the eigenvectors $\underline{\phi}'_1, \underline{\phi}'_2, \ldots, \underline{\phi}'_M$ are orthornormal (Strang, 1980). Thus, the features y'_m in the first-stage output random (feature) vector $\mathbf{y}'_M = [\mathrm{y}'_1 \cdots \mathrm{y}'_m \cdots \mathrm{y}'_M]^T$ are uncorrelated, i.e. $E(\mathrm{y}'_i\mathrm{y}'_j) = 0$ for $i \neq j$.

In the second stage, we nonlinearly transform the reduced M dimensional output feature random vector $\mathbf{y}'_M$ to the power p to produce the higher-order features $\mathbf{y}'_{Mp} = [\, y_1'^p \quad y_2'^p \quad \cdots \quad y_M'^p\,)^T$, and then compute a second transform $\underline{\Phi}_K$ to yield the *final nonlinear output feature space* of dimension K, $\mathbf{y}_K = \underline{\Phi}_K^T\mathbf{y}'_{Mp}$. Note that the nonlinearities introduced in the two stages are different (p' in stage 1, and p in stage 2); this is expected to produce more general nonlinear transforms on the input image data. We choose M features (or basis functions) in stage 1 to be smaller than the number of training sample data S, i.e. $M \leq S$. This allows us to use the standard E_D performance measure in stage 2, since its sample covariance matrix $\underline{\mathrm{C}}_p$ can be inverted easily. The second stage transform $\underline{\Phi}_K$ we use maximizes the measure in Eq. (2) applied to the new $\mathbf{y}'_M$ random vector. Specifically, the transform $\underline{\Phi}_K$ maximizes the discrimination measure $E_\mathrm{D} = [\underline{\Phi}_K^T\underline{R}_{12p}\underline{\Phi}_K]/[\underline{\Phi}_K^T(\underline{C}_{1p} + \underline{C}_{2p})\underline{\Phi}_K]$, where $\underline{\mathrm{C}}_{1p}$, $\underline{\mathrm{C}}_{2p}$, and $\underline{\mathrm{R}}_{12p}$ are the intra-class covariance and inter-class

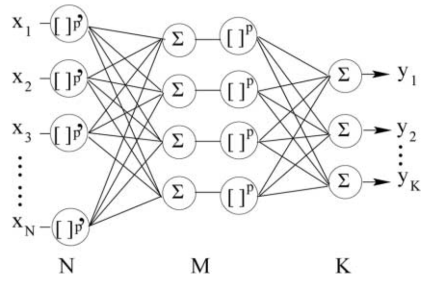

Fig. 3. Neural network implementation of the nonlinear MDFs for high-dimensional inputs.

difference correlation matrices of the reduced M-dimensional feature data $\mathbf{y}'_{Mp}$. The set of transformations $\underline{\Phi}_K$ are, by analogy to Eq. (4), the dominant eigenvectors of

$$[\underline{C}_{1p} + \underline{C}_{2p}]^{-1}\underline{R}_{12p}. \tag{7}$$

Since the feature space $\mathbf{y}'_M$ (and $\mathbf{y}'_{Mp}$) has a dimensionality $M \le S$, the inversion of the matrix in Eq. (7) is trivial.

Note that our first stage measure does not minimize the intra-class spread of class projections; however, this is done in the second stage. We also note that the number of on-line computations to obtain K output MDF features in the second stage with M features in stage 1 is

$$3\mathcal{O}(N) + M\mathcal{O}(N) + \mathcal{O}(M \times K) \ll \mathcal{O}(N) \simeq M\mathcal{O}(N). \tag{8}$$

which follows from Eq. (6). Thus, our new two-stage nonlinear algorithm has a *closed-form solution and has on-line computational complexity of order N*.

This two-stage nonlinear MDF transform for image input data can be expressed as a neural network with polynomial activation functions. Fig. 3 shows its implementation for high-dimensional inputs with N dimensions, M features in the first stage, and K output features in the second stage. The neural network implementation uses N input neurons with polynomial activation functions (p'), M summation neurons in the hidden layer with polynomial activation functions (p), and K neurons in the output layer.

3.1. Rank and other properties of the two-stage MDF transformations

We now discuss some aspects of the two-stage MDF for high-dimensional inputs. Specifically, we will detail the variety of the types of transformations or decision surfaces it can produce in terms of the rank, principal orientation, and degree of the transform. We analyze nonlinear MDF transforms with only *integer-valued polynomials*; polynomial transforms with real-valued polynomial orders are hard to analyze mathematically and, hence, are not discussed here. Special attention is paid to quadratic two-stage transforms, since their properties are well known and easy to analyze.

We first show that the two-stage MDF can produce transforms with any principal orientation and that it can yield full-rank transforms that curve in all N input dimensions. However, it *cannot produce both full-rank transforms with any principal orientation simultaneously*. We analyze and prove this only for the quadratic case.

We first explicitly write the k-th output feature from the second stage of the MDF neural network for a given input data vector $\underline{x}$ with any polynomial orders p' and p. The first stage transform is $\underline{y}'_M = \underline{\Phi}'^T_M \underline{x}_{1p'}$. The nonlinear transform of the features from the first stage using the transform matrix $\underline{\Phi}_K$ yields a k-th nonlinear output feature, $y_k = \underline{\phi K}^T_k \underline{y}'_{Mp} = \sum_{m=1}^{M} \phi_{mk} y'^p_m$. Since $y'_m = \sum_{j=1}^{N} \phi'_{jm} x_j^{p'}$ for $m = 1, 2, \ldots, M$, the k-th output feature for the two-stage nonlinear MDF has the general form

$$y_k = \sum_{m=1}^{M} \left[\phi_{mk} \left(\sum_{j=1}^{N} \phi'_{jm} x_j^{p'} \right)^p \right]. \tag{9}$$

The transformation in Eq. (9) *contains cross-product terms* of the input pixels x_j. For example, with $p' = 1$ and $p = 2$, the k-th output MDF feature is $y_k = \sum_{m=1}^{M}[\phi_{mk}(\sum_{i=1}^{N}\sum_{j=1}^{N}\phi'_{jm}\phi'_{im}x_j x_i)]$.

The presence of the cross-product terms allows the nonlinear MDF output transformations to have *any principal orientation* as we now show. When $p' = 1$, the k-th output MDF feature is

$$y_k = \sum_{m=1}^{M} \phi_{mk}(\underline{\phi}'^T_m \underline{x})^p. \tag{10}$$

This involves a sum of M linear transforms (VIPs) of the input data with M transformation vectors $\underline{\phi}'_m$, where each VIP is raised to a polynomial power p. Eq. (10) can then be written as a weighted sum of nonlinear functions as $y = \sum_{m=1}^{M} a_m f(\underline{w}_m^T \underline{x})$. We prove in Appendix A that such a nonlinear transform $a_m f(\underline{w}_m^T \underline{x})$ curves in only one dimension in the input space, with a principal orientation along $\underline{\phi}'_m = \underline{w}_m$. Therefore, M such linear combinations have a principal orientation in the subspace spanned by the M transformation coefficients $\phi'_1, \phi'_2, \ldots, \phi'_M$. Therefore, *an MDF with $p' = 1$, and any p can have any principal orientation* because of the cross-terms in Eq. (10). Its principal orientation is determined by the subspace spanned by the M transformation coefficients $\phi'_1, \phi'_2, \ldots, \phi'_M$. The rank of such a transform is only M, since each term in the summation in Eq. (10) is of rank one as proved in Appendix A.

The two-stage MDF can yield full-rank (N) transformations. We now prove this for the quadratic case only ($p' = 2$ and $p = 1$). The k-th MDF output feature in this case is

$$y_k = \sum_{m=1}^{M} \left[\phi_{mk} \left(\sum_{j=1}^{N} \phi'_{jm} x_j^2 \right) \right]. \tag{11}$$

It can be written in matrix notation as

$$y_k = \sum_{m=1}^{M} [\phi_{mk}(\underline{x}^T \underline{A}_m \underline{x})], \tag{12}$$

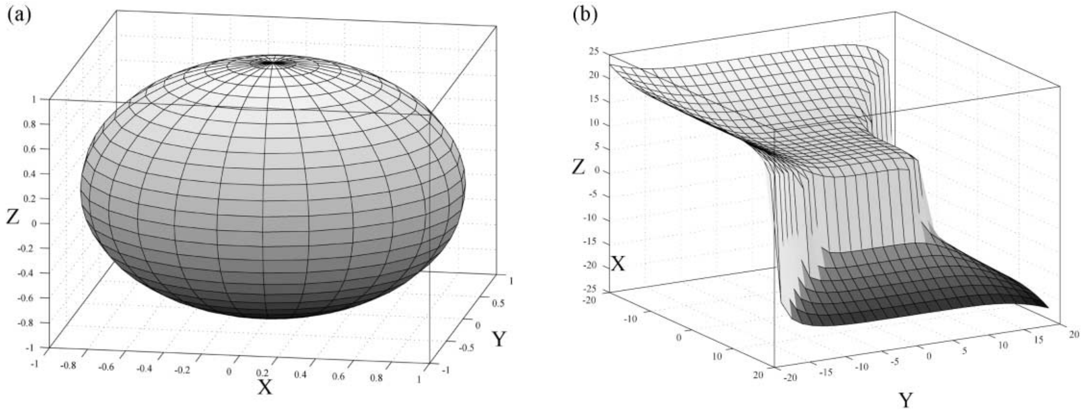

Fig. 4. Loci for the transformations from the first stage of the two-stage MDF for 3-D inputs produced with different p' first-stage polynomial powers. (a) $p' = 2$. (b) $p' = 5$.

which is a linear weighted sum of M quadratic transforms. Each quadratic transformation matrix $\underline{A}_m$ is an $N \times N$ diagonal matrix whose diagonal elements are the first stage MDF transformations coefficients $\underline{\phi}'_m$. The rank of each $\underline{A}_m$ is determined by the number of non-zero MDF coefficients in $\underline{\phi}'_m$. Therefore, *the maximum rank of the quadratic transformation matrix* $\underline{A}_m$ *is N* if *all* of the corresponding MDF coefficients in $\underline{\phi}'_m$ are non-zero. We now consider the rank of one output MDF feature in Eq. (12), which is a linear combination of M quadratic transforms, each of which can be of full rank. Since $\sum_{m=1}^{M} \phi_{mk}\underline{x}^T\underline{A}_m\underline{x}$ is the same as $\underline{x}^T(\sum_{m=1}^{M} \phi_{mk}\underline{A}_m)\underline{x}$, we can write (Strang, 1980)

$$\mathrm{rank}\left(\sum_{m=1}^{M} \phi_{mk}\underline{A}_m\right) \le \min\left\{N, \sum_{m=1}^{M} \mathrm{rank}(\phi_{mk}\underline{A}_m)\right\}$$

$$= \min\left\{N, \sum_{m=1}^{M} \mathrm{rank}(\underline{A}_m)\right\}. \qquad (13)$$

The maximum rank of the nonlinear transformation can, thus, be N, since each $\underline{A}_m$ can be of full rank as discussed earlier, and can curve in all input dimensions. However, the quadratic transformation matrix $\sum_{m=1}^{M}(\phi_{mk}\underline{A}_m)$ does not contain any cross-product terms. Therefore, *the principal orientation can only lie along the input axes*.

We now discuss the degree of the transformations produced by our two-stage MDF algorithm. In the two-stage MDF, the effective polynomial order of the transformation is $p' \times p$ as we see from Eq. (9). The optimal polynomial p' and p values are determined automatically from training data for a specific application as discussed later in Section 5. In order to visually illustrate the effects of different polynomial orders on the decision boundaries, we show 3-D input data examples in Fig. 4 with $p = 1$ and $M = K = 1$. In each figure, the loci of nonlinearly related input data that are transformed to the same MDF output feature value are shown as a nonlinear surface in the 3-D input space. Note that closed decision boundaries can be obtained with a quadratic transform ($p' = 2$ in Fig. 4a). A fifth-degree polynomial transform (Fig. 4b) produces a more complex decision boundary than a linear one. We show in Section 5 that such higher-degree nonlinear transforms are needed for some discrimination problems.

3.2. Other theoretical aspects of the two-stage MDF transformations

We now show that our new two-stage nonlinear transformation technique specializes to some well-known image processing methods for specific choices of the nonlinearities p and p'. For $p' = p = 1$, the operations in both stages of the two-stage MDF are linear and, thus, *our nonlinear transforms for images can emulate linear filtering operations at the image center*. For high positive and negative values of p', some other interesting transforms are obtained, as we discuss next.

We show in Appendix B that for the case when $\underline{\Phi}_K$ is a diagonal identity matrix (with $K = M$), and $p = 1/p'$, our nonlinear MDF is exactly the same as the well-known weighted p-norm operation. The weighted p-norm has been used for image classification using ideas from fuzzy morphology (Gader, 1992), where both the coefficients ϕ_m and the nonlinearity p are determined iteratively. We show in Appendix B that for the case when $p' \to -\infty$ (or $p' \ll 0$) the output nonlinear MDF feature for a given data sample image $\underline{x}$ is $y = \min(\{x_i\})$, which corresponds to the *mathematical morphology operation of erosion* (Dougherty, 1986) over the i image pixels corresponding to the nonzero MDF coefficients $\{\phi_i\}$. Therefore, in this case, the MDF NN weights (transformation coefficients) $\underline{\phi}_m$ correspond to a morphological structuring element whose size is the same as the input image size. The 'on' pixels in the structuring element are the non-zero MDF coefficients, and the 'don't-care' pixels are those with zero (or close to zero) MDF coefficient values (Dougherty, 1986). When $p' \to +\infty$ (or $p' \gg 0$), we prove in Appendix B that the max operation is approximated by our MDF, which is the *morphology operation of dilation*. Note that more complex morphological

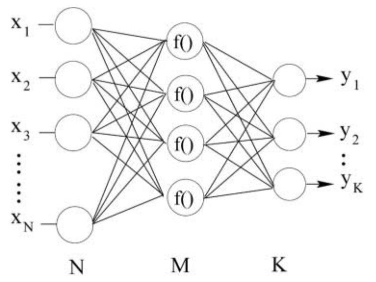

Fig. 5. A neural network for nonlinear PCA feature extraction with one hidden layer.

operations such as closing (dilation followed by erosion) and opening (erosion followed by dilation) can also be implemented by using two two-stage MDF operations in sequence. *This generality of our transform to various well-known and diverse image processing operations makes it applicable to a variety of image processing and feature extraction problems*. We will show experimentally in Section 5 that the nonlinear MDF automatically selects the correct values of the polynomials to emulate the max, min, and quadratic operations when needed.

4. Other nonlinear methods for high-dimensional inputs

In this section, we compare the rank, degree and principal orientation of the transformations produced by the two-stage MDF (Section 3) with those for other relevant techniques on *high-dimensional input data only* (large N where $S < N$, with the additional constraint that $S \ll N$ for image inputs). The first methods we consider are neural networks for dimensionality reduction and feature extraction (Baldi & Hornik, 1989; Kramer, 1991) including nonlinear principal component analysis (nonlinear PCA) (Karhunen & Joutsensalo, 1994, 1995). We consider these neural network methods since they perform nonlinear transforms for dimensionality reduction or feature extraction of high-dimensional inputs. The second method we consider uses kernel-based techniques for feature extraction (Scholkopf et al., 1998) and classification (Cortes & Burges, 1998; Vapnik, 1995). We analyze these kernel-based methods since, like the MDF approach, they note that one can linearize a nonlinear transformation to solve for the nonlinear mapping coefficients (Scholkopf et al., 1998) and they discuss polynomial transformations on input data that are similar to those in the nonlinear MDF. Both nonlinear PCA neural networks and kernel-based methods implement transforms that involve linear weighted combinations of the input data passed through a nonlinearity. We now show that such nonlinear transformations have limited rank and a fixed degree compared to the MDF, which yields full-rank transformations (Section 3.1) and with variable degree (neural network activation functions) that is automatically selected. The technique we use to select the degree is discussed in Section 5. We first describe the nonlinear PCA neural networks (Section 4.1) and kernel-based techniques (Section 4.2); we then (Section 4.3) discuss the form of their nonlinearities and their transformations in terms of their rank, principal orientation, and degree in a unified manner.

4.1. Nonlinear PCA neural networks for dimensionality reduction

Neural networks for dimensionality reduction of high-dimensional input data have the constraint that the number of neurons in the hidden layers should be less (and for the case of image inputs *significantly* less) than the input dimension N. These neural networks are referred to as nonlinear PCA neural networks in the signal processing and statistical community. Most prior work on nonlinear PCA neural networks considered a Hebbian-based neural network (Karhunen & Joutsensalo, 1994, 1995) with a single layer or with two layers or with two layers (one hidden layer), and a multilayer perceptron autoassociative neural network (Baldi & Hornik, 1989; Kramer, 1991). We shall only analyze the first case, since they are more commonly used for dimensionality reduction. The architecture of a neural network for dimensionality reduction of N dimensional input data is shown in Fig. 5, with N input neurons, M hidden layer neurons, and K output feature neurons. In these nonlinear PCA neural networks for feature extraction (and most other neural networks, including the multilayer perceptron), the first (input) layer is transparent with a *linear* activation function $f(x) = x$. For dimensionality reduction with large N, we require M and $K \ll N$. The $M \ll N$ second layer neurons have nonlinear activation functions $f(\,)$ (typically sigmoids, tanh, etc.), and have M inter-connecting weight vectors $\underline{w}_m$ (from each input to the M neurons in the hidden layer). The m-th hidden layer neuron's output is $f(\underline{w}_m^T\underline{x})$. We will show that these hidden-layer neurons have a significant effect on the performance of neural networks on high-dimensional inputs when $M \ll N$. The K output neurons have *linear activation functions* and yield the final $K < N$ output features. For the general two-layer nonlinear PCA neural network in Fig. 5, the output at the k-th output neuron can be written as

$$y_k = \Psi(\underline{x}) = \sum_{m=1}^{M} w_{mk} f(\underline{w}_m^T\underline{x}), \tag{14}$$

where w_{mk} is the weight between the m-th hidden layer neuron and the k-th output layer neuron.

Our two-stage MDF neural network in Fig. 3 is similar to that of the nonlinear PCA neural network (Fig. 5), except that the activation functions at the input and hidden-layer neurons are polynomial powers (they can be different in the two layers). In the MDF neural network, the neuron activation functions are adapted; in nonlinear PCA neural networks they are fixed (sigmoid, tanh, etc.) and nonlinear functions are only present in the hidden layer.

Another neural network approach to nonlinear PCA (Baldi & Hornik, 1989; Kramer, 1991) uses the architecture of Fig. 5 with $K = N$ output neurons during training and autoassociative learning (the desired output is the input). After training, the output layer is discarded and the M hidden layer outputs are the reduced-order nonlinear PCA features. Both of these nonlinear PCA neural network dimensionality reduction techniques have very similar architectures after completion of training. Use of nonlinear error measures (Karhunen & Joutsensalo, 1995) and sigmoid, tanh, etc. activation functions (Karhunen & Joutsensalo, 1994, 1995) ensures use of higher-order statistics, and nonlinear transformations of the input. However, *nonlinear PCA neural networks are intended for representation and not for discrimination.*

4.2. Kernel-based nonlinear techniques

Kernel-based methods are categorized into nonlinear kernel PCA representation (Scholkopf et al., 1998) and support vector machines (SVMs) for classification (Burges, 1998; Cortes & Vapnik, 1995). Kernel PCA is a nonlinear extension of the linear PCA method. For the high-dimensional case being considered, the authors of kernel PCA note that because of the computation problems noted in Section 3, solutions using all explicit higher-order cross-terms of the input data are not realistic. Instead, they consider the output feature to be of the form

$$y = \sum_{i=1}^{S} \alpha_i \mathscr{K}(\underline{w}_i, \underline{x}), \tag{15}$$

where the kernel function $\mathscr{K}(\underline{w}_i,\underline{x})$ is typically a linear combination of the input data $\underline{x}$ with 'weights' $\underline{w}_i$ (these weights are the training data vectors $\underline{x}_i$) followed by a nonlinear mapping. Thus, the kernel-based PCA outputs in Eq. (15) are of the same form as Eq. (14) for nonlinear PCA neural networks with the summation over S, not M. The advantage of kernel PCA is that the nonlinear solutions are obtained in closed-form while ensuring low on-line computational complexity through the use of kernels instead of explicit higher-order mappings. Polynomial kernel-based PCA methods (Osuna, Freund & Girosi, 1997; Scholkopf et al., 1998) are quite similar to the polynomial MDF transforms; they use kernels of the form $\mathscr{K}(\underline{w}, \underline{x}) = (1 + \underline{w}^T \underline{x})^p$, where p is the polynomial order and is typically chosen to be a positive integer. However, *these kernel PCA methods are only intended for representation*; classification is of more concern to us.

Support vector machines (SVMs) (Vapnik, 1995) for *classification* involve designing classifiers based on only a few so-called support vectors that lie close to the decision boundary between two classes. *Linear* SVM calculates a linear basis function $\underline{w}$ that maximizes the minumum distance between classes. It is (Osuna et al., 1997; Vapnik, 1995) a linear combination of the V training vectors (support vectors) $\underline{x}_i$ that lie close to the samples in the other class, $\underline{\phi} = \sum_{i=1}^{V} a_i z_i \underline{x}_i$, where the a_i are the solutions for the non-zero Lagrange multipliers of a quadratic cost function and the z_i are the desired output value for the i-th sample. Therefore, the SVM output for an unknown test input $\underline{x}$ is $y = \underline{\phi}^T \underline{x} = \sum_{i=1}^{V} a_i z_i \underline{w}_i^T \underline{x}$ where we denote the support vectors $\underline{x}_i$ as $\underline{w}_i$ for consistency. Nonlinear SVMs use nonlinear kernel mappings of the input data $\mathscr{K}(\underline{x},\underline{w}_i)$, similar in concept to kernel PCA (Eq. (15)), to yield nonlinear outputs

$$y = \sum_{i=1}^{V} a_i z_i \mathscr{K}(\underline{x}, \underline{w}_i), \tag{16}$$

for input data $\underline{x}$ with the summation over the V support vectors. A commonly used polynomial kernel is $\mathscr{K}(\underline{w}, \underline{x}) = (1 + \underline{w}^T \underline{x})^p$ as used in kernel PCA. Unlike the kernel-based PCA solution, however, the a_i Lagrange multiplier solutions in Eq. (16) in the SVM are obtained iteratively (Osuna et al., 1997; Vapnik, 1995). Solving for the Lagrangians a_i in the SVM is a minimization problem involving a quadratic form matrix (Burges, 1998; Osuna et al., 1997) that is completely dense and of size $S \times S$. This implies that when the training data set S is large ($\geq$5000) (Osuna et al., 1997), this problem cannot be solved on a PC or equivalent computer without data and problem decomposition (Osuna et al., 1997). Another issue in the SVM is that when the data classes overlap, a set of slack-variables are used that measure the amount of misclassification in the SVM (Osuna et al., 1997). In such a case, the SVM requires a user-defined 'cost-parameter', which is a measure of the amount of misclassifications that will be tolerated (Burges, 1998; Cortes & Vapnik, 1995; Osuna et al., 1997). We now discuss the properties of the nonlinear transformations produced by these other nonlinear transformation methods.

4.3. Theoretical properties of transformations by nonlinear PCA and polynomial kernel-based methods

4.3.1. Ranks of nonlinear PCA and kernel-based methods

We will now derive the ranks of the nonlinear transforms produced by nonlinear PCA neural networks, polynomial kernel PCA, and polynomial SVMs. Note that the nonlinear transformations produced by two-layer nonlinear PCA neural networks (Eq. (14)), polynomial kernel PCA (Eq. (15)), and polynomial SVMs (Eq. (16)) *have the same form with the main difference being the number of terms in each summation* (M, S, and V, respectively). All of these methods involve the summation of linear weighted combinations of the input $\underline{x}$ passed through a nonlinearity. In Appendix A, we prove that each nonlinear transform of the form $f(\underline{w}^T\underline{x})$ is of rank one, regardless of the mapping $f(\,)$ used. We also show in Appendix A that M linear combinations (weighted summations) of such transforms would curve in *at most M*input dimensions. Therefore, *nonlinear PCA neural networks* (Eq. (14)) *produce nonlinear transforms of maximum rank M* (since M hidden layer neurons are present). We feel that nonlinear transforms produced by neural networks with more hidden layers (>1) also have rank M. However, we do not have a rigorous proof for this. *These three other nonlinear techniques noted, thus, produce transforms of rank M, S, and V, respectively.* Note that since M, S, and V are all $\ll N$ for high-dimensional inputs, *none of these other nonlinear techniques can produce closed surfaces* (transforms that curve in all N input dimensions). The two-stage MDF can produce nonlinear transforms that can form piecewise-closed surfaces in the input space (Section 3). We note that support vector machines that use other kernel functions, such as radial basis function kernels, could produce full-rank nonlinear transforms since they involve a weighted sum of exponentials of the squares of input data and the training samples, similar to a weighted sum of quadratic transforms that our nonlinear MEF implements with a second-order polynomial in the first layer (stage).

We have not seen any prior theoretical comparison of the *ranks* of these different nonlinear transforms. Most prior work on SVMs (Burges, 1998; Osuna et al., 1997; Scholkopf, Burges & Vapnik, 1995) involved cases with $S > N$, where the training set size exceeds the input dimensionality. We have not found *any* prior reference that theoretically discusses the effects of the types of transformations with varying training set sizes. Malthouse (1998) proved that nonlinear PCA transformations cannot produce closed surfaces. His proof assumed smooth transfer functions $f(\,)$ in the hidden layer; our proof holds for *any* transfer function $f()$, including threshold transfer functions. We note that for the nonlinear PCA neural network, if M is increased (more nonlinear PCA features are used), the number of neural network weights increases and so does the size of the required training set. For dimensionality reduction, $M \ll N$ is needed.

4.3.2. Principal orientation of nonlinear PCA and kernel-based methods

We now address the principal orientation each nonlinear transform allows. We showed earlier (Section 3) that the two-stage MDF can yield transforms with any principal orientation. Nonlinear PCA neural network transformations curve in the subspace spanned by the second layer weight vectors $\mathrm{W} = [\underline{\mathrm{w}}_1\ \underline{\mathrm{w}}_2\ \cdots\ \underline{\mathrm{w}}_m\ \cdots \underline{\mathrm{w}}_M]$, and therefore, the principal orientation also occurs in this subspace. Transforms by polynomial kernel PCA have principal orientation lying in the subspace spanned by the S training data samples $\underline{\mathrm{x}}_i$; principal orientations of the polynomial SVM transformations lie in the subspace spanned by the V support vectors. Therefore, these nonlinear techniques can have transforms with *any principal orientation* in the subspace spanned by the M weights (for nonlinear PCA), by the transformation vectors $\underline{\mathrm{w}}_1$, $\underline{\mathrm{w}}_2$, …, $\underline{\mathrm{w}}_S$ (for polynomial kernel-based methods), and by the $\underline{\phi}_1$, $\underline{\phi}_2$, …, $\underline{\phi}_M$ (for the nonlinear MDF algorithm).

4.3.3. Degree of various nonlinear methods

The degree of the different nonlinear transformations is now addressed. The degree for nonlinear PCA neural networks and polynomial kernel-based methods (including polynomial SVMs) are fixed in advance (since the neuron activation function or the form of the polynomial kernel function are fixed), whereas the degree of our two-stage MDF transforms can be anything and is determined automatically based on training set data as the values that maximize our new discrimination measures E_{D} and E'_{D}. We expected this property of the two-stage MDF neural network to be advantageous compared to fixed-degree transformations. Since the activation functions used in the nonlinear PCA neural networks are continuously differentiable up to any order, it is expected that nonlinear PCA neural networks could yield transforms of any degree. The degree of the transformations for polynomial kernels (Cortes & Vapnik, 1995; Scholkopf et al., 1998) is p and is similar to that of the nonlinear MDF (Section 3), but is fixed in advance and is not automatically selected based on the training data.

4.3.4. Computational complexity of various nonlinear methods

The training (off-line) complexity of the different nonlinear methods varies considerably. The weights in the neural networks for dimensionality reduction are computed iteratively. The solution for the SVM weights involves an iterative quadratic optimization scheme. It is well known that most iterative schemes are slow and have associated convergence problems (Casasent et al., 1998). Both the MDF and kernel PCA have a closed-form solution and, hence, the training time is not an issue.

We now address the on-line computation times for the different approaches. The number of on-line computations for nonlinear PCA neural networks is proportional to $M\mathcal{O}(N)$, where a VIP requires $\mathcal{O}(N)$ operations; kernel PCA features require $S\mathcal{O}(N)$ computations, and SVM outputs require $V\mathcal{O}(N)$ computations. Each two-stage MDF nonlinear feature can be computed in $\simeq M\mathcal{O}(N)$ time (Eq. (8)). Typically, $V \simeq M$ and, thus, the on-line computation times for nonlinear PCA, SVMs and MDFs are comparable and are much less than those for kernel PCA. In some cases, the number of support vectors V can become large ($\simeq 1000$ in an application involving face detection, Osuna et al., 1997). Therefore, kernel-based techniques *could* have a greater on-line computational load compared with MDFs and nonlinear PCA neural networks.

5. Results

We now address the performance of the two-stage MDF neural network for several cases *involving data classification (discrimination)*. The performance of the MDF neural network for discrimination (classification) is compared with that of a standard multilayer perceptron (MLP) neural network, linear Fisher linear discriminant and ODV discriminant methods, and with that of polynomial support vector machines. PCA methods are not considered as they do not address discrimination/classification (we have shown earlier, Talukder & Casasent, 1998, 1999, that our MDF neural network output features designed for discrimination are superior to PCA ones for problems involving classification). *Special attention is paid to the high-dimensional input data case. In order to adequately simulate the high-dimensional case, we constrain the number of hidden-layer and output-layer neurons (M and K) in the MDF and MLP neural networks to be less than the input dimensionality N, i.e. M, K < N in our experiments*. We consider two cases: (1) where the number of training samples $S \geq N$, the dimensionality of the input data; and (2) the high-dimensional case where $S < N$. We theoretically proved earlier that standard neural networks, and polynomial kernel-based methods such as the polynomial support vector machine provide limited nonlinear transforms for high-dimensional input data where $S < N$. We will verify this through experiments in this section. Most prior experimental results using polynomial-kernel SVMs (Burges, 1998; Osuna et al., 1997; Scholkopf et al., 1995) on synthetic datasets (Burges, 1998) used $S \gg N$, and typically involved only 2-D inputs ($N = 2$) for purposes of visualization. Prior work on SVMs with real datasets including the SVM face detection system (Osuna et al., 1997) used $S = 5000$–50,000 training samples with an input dimensionality $N = 283$ face pixels, while the SVM character recognition system (Scholkopf et al., 1995) used $S = 7300$ with input images consisting of $N = 16 \times 16 = 256$ pixels. Very little work has been done using SVMs with $S < N$; Roobaert and Hulle (1999) considered multi-view object classification of $N = 32 \times 32 = 1024$ pixel input images with a training set sample size of $S = 4$ per class.

We consider two-dimensional (2-D) and fifty-dimensional (50-D) input data cases. For the case where $S \geq N$ (Section 5.1), we use only 2-D inputs and $M = K = 1$ neurons in the MDF and MLP neural networks. *One purpose in using 2-D inputs is for visual illustration of the nonlinear transforms* produced by the MDF neural network and the SVM classifier. For the case where $S < N$ (Section 5.2), we use 50-D input data. In all these synthetic case studies, we consider two input classes where class 1 samples are marked as '○' and class 2 samples as ' + ', with equal prior probabilities for each class. For the different 2-D case studies, we show the output feature transformations (mappings) as contour plots; in the MDF plots, each line (contour) corresponds to the set of points in the input data that map to the same output feature value. In the SVM plots, the SVM MATLAB plotting routine shows the contours as different gray-scale shaded regions. Since all our cases are discrimination applications, the decision boundary is parallel to the contour lines (for linear transformations) or curves (for nonlinear transformations).

We used the following performance measures to evaluate results. For all 2-D input cases, we evaluate performance of a technique using a *visual criterion* and the mathematical measure P_C (probability of correct classification) of the test samples.

We also analyze the on-line computation time for each method. For MDF and multilayer perceptron neural networks, the computation time is dependent primarily on the number of neurons M used in the first hidden layer (Eq. (8)); for SVMs, it is dependent on the number of support vectors V.

MDF neural network for discrimination

Since the purpose is to test the performance of the MDF neural network (NN) for high-dimensional inputs, we used the two-stage nonlinear MDF (Section 3). For 2-D inputs, we used only $M = K = 1$ neurons in the MDF neural network. For 50-D input data, we varied $M = K$ from 1 to 41 in increments of 10.

We used an exhaustive search method to pick the p' and p values in the two-stage nonlinear MDF neural network for a specific application. The nonlinear MDF solution for each p' and p is obtained (from the training set) in closed-form by searching through the p' and p range (± 20) in increments of 0.1 for polynomial values between -3 and $+3$, and in coarse increments for higher polynomial values between $+3$ and $+19$, and -3 and -19. Note that we use real-valued and negative-valued polynomials; this is expected to give more general transformations than positive-integer-valued polynomials. The value of the p' and p pair that gave the best discrimination E_D for $M = K = 1$ for the training set was selected. This search is fast and the exact p' and p choices are not critical.

Multilayer perceptron for discrimination

The MLP NN architecture we compare against our MDF NN for high-dimensional inputs had two hidden layers; it has been shown (Bishop, 1995, p. 123) that this can produce disjoint non-convex closed decision boundaries. Since the purpose of this case study is to simulate high-dimensional inputs, for which dimensionality reduction is needed, we constrained the number of hidden layer neurons M in the second and third layers to satisfy $M < N$; we used equal number of neurons in both hidden layers. The number of neurons in the output layer was equal to the number of classes, two; each input sample is classified using a winner-take-all criterion at the output layer. The learning rule used to train the multi-layer perceptron was the Levenberg–Marquardt algorithm (Bishop, 1995) that approximates the Hessian (second-order derivative) matrix while constraining the step size in the weight update to be small. We found this to give better P_C performance than other methods when implemented in the MATLAB Neural Network Toolbox. The number of training epochs was set to 10^5 iterations in batch learning mode, with the maximum training time set to 10 h. All neural network weights were randomly initialized to values in $[-0.1, 0.1]$. The default values were used for the other parameters in the neural network learning rule. Prior work (Denoeux & Lengelle, 1993) noted that random weight initialization resulted in training problems with multilayer perceptron neural

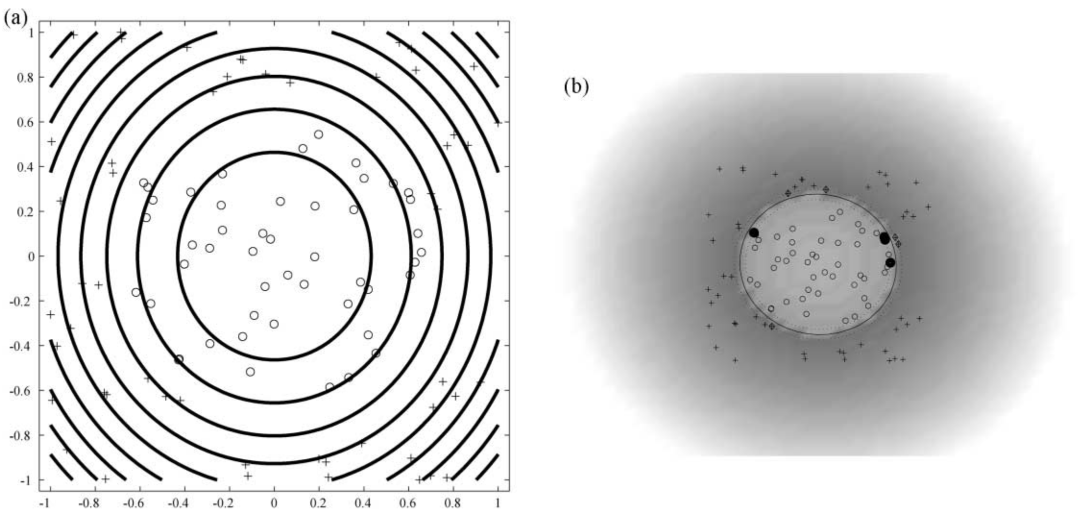
Fig. 6. (a) 1-D nonlinear MDF neural network contour path: Contour plots for 2-D input data with circularly-distributed classes superimposed on the training data samples using 1-D nonlinear MDF output features, and (b) SVM contour plot: the corresponding SVM decision boundary shown as a dark line with shaded regions representing SVM output contours.

networks, generalization problems, and convergence to local minima in some cases. We noticed these problems with the multilayer perceptron in our synthetic data case studies.

SVMs for discrimination

The SVM software written in MATLAB code by Steve Gunn was used (www.isis.ecs. soton.ac.uk/resources/svminfo/). We used the polynomial kernel SVM for comparison purposes, since it is closer to our MDF neural network than are other kernels. The parameters associated with the SVM polynomial kernel method are: the cost function C that signifies tolerance to misclassification errors (higher values imply less tolerance); and the polynomial order. After experimentation, we found that $C = 100$ yielded good performance on 2-D input data. For lower values of C, P_C performance was poorer, and for higher values of C overfitting occurred (because all/most training data samples were correctly classified); a similar trend was observed by others (Gunn, 1997).

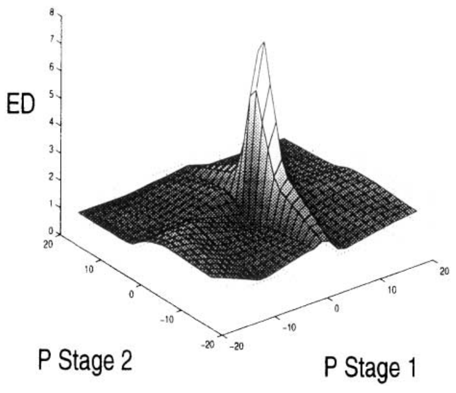

Fig. 7. E_D versus p' and p for the circularly-distributed 2-D input data in Fig. 6.

The polynomial order used was also selected using a visual criterion. Polynomial orders from two to 19 were tested.

5.1. Case studies with $S \geq N$

In this section, we evaluate the performance of the different methods on 2-D ($N = 2$) cases where the number of training samples is greater than the input dimensionality $S \geq N$. We used two synthetic test cases that are *not* linearly separable. We used a total of $S = 100$ training samples ($\simeq$50 per class) in all 2-D cases. A larger number of $S = 2000$ training samples improved P_C test data performance slightly in the case studies, but was not used due to the prohibitively large training time of the SVM which involves an optimization using a matrix of size $S \times S$. Test set performance was based on 2000 test samples ($\simeq$1000 per class) to provide statistically significant P_C measures.

5.1.1. Case study—1 (circularly-distributed data)

The first data set is one in which one class enclosed the other class samples within a *circle*. Samples were generated with a uniform random distribution in [-1, 1]; those within a radius of 0.5 from the origin were assigned to class 1 and all others were assigned to class 2 as shown for the training samples in Fig. 6a. A gap of 0.03 was present between the two classes in training and test set data (Fig. 6). For this class distribution, the optimal discriminant is a quadratic transformation of the input data. The nonlinear MDF can implement a quadratic transform if $p' = 2$, and $p = 1$. The E_D discrimination measure in the MDF as p' and p were varied is shown in Fig. 7. It peaked at $p' = 2$ and $p = 1$; this choice was selected automatically and it produces an exact quadratic transform which is the optimal discriminating function in this case. The P_C for our nonlinear MDF is

Table 1
P_C and computation time for the circularly-distributed 2-D data on the test set using different methods

	MDF NN	MLP NN	MLP NN	FLD	SVM
	$(M, K = 1)$	$(M = 1)$	$(M = 2)$	(1)	$(V = 9)$
P_C (%)	100	66.4	100	65.2	100
Comp. time	1	1	2	1	9

therefore perfect (100%). Fig. 6a shows the MDF contour plots superimposed on the 100 training samples. the SVM output contours using a second-order polynomial kernel are shown as shaded regions and the decision boundary is shown as a solid line superimposed on the 100 training data samples in Fig. 6b; the support vectors are the filled-in circles close to the decision boundary. The test set P_C performance of the different methods on the test data samples is shown in Table 1. The MDF and SVM performance was perfect. The multilayer perceptron neural network (MLP NN) performed worse because it used only $M = 1$ hidden-layer neuron and, therefore, can curve in only one dimension in the input space as we proved earlier. Therefore, it cannot produce a closed decision surface. For $M = 2$ hidden-layer neurons, the multilayer perceptron also gave perfect P_C performance because closed surface transformations are obtained with two hidden-layer neurons on 2-D input data, as we proved theoretically in Section 4. The Fisher linear discriminant (FLD) also gave poor P_C since it can only perform linear transforms of the input data. The on-line computational complexity (computation time) of the SVM classifier is the highest (9), whereas the other methods are significantly faster (Table 1, bottom row); this occurred because the SVM algorithm chose $V = 9$ support vectors.

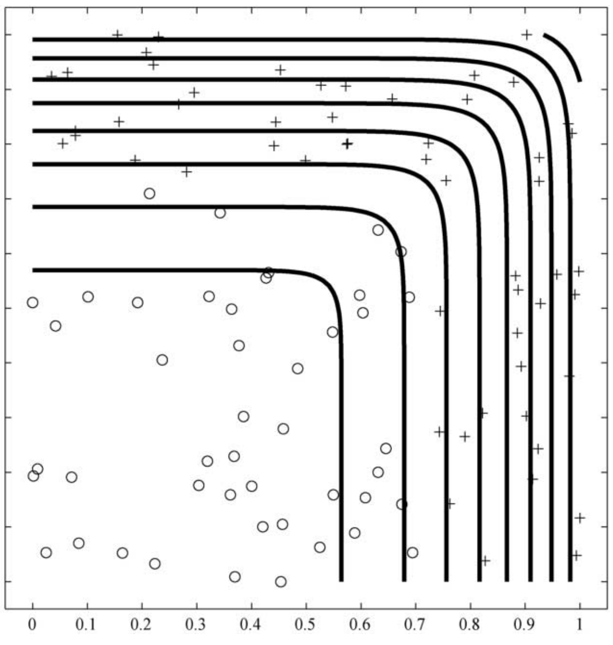

Fig. 8. 2-D max-distributed input training data superimposed on the 1-D nonlinear MDF contour plots.

5.1.2. Case study—2 (max function)

The second 2-D synthetic data set considered is one in which the class label is determined by a max function. Given a sample $\underline{x}$ of dimension N, it is a member of class 1 if max($\underline{x}$) is smaller than 0.75, else it is a member of class 2. This ensures approximately an equal number of samples in both classes. The max of an $N \times 1$ vector is defined as the maximum of *all* of its N elements. Fig. 8 shows the training data samples for the $N = 2$ dimensional case. A gap of 0.03 is present between the two classes (Fig. 8) for both training and test data. Samples with such class distributions are not linearly separable; hence, a nonlinear feature extraction procedure is required. In fact, the optimal discriminant function would result in a mapping of the form $y = \max(\underline{x})$. A max function is a nonlinear mapping that must curve in all directions in the input space; hence, we do not expect multilayer perceptron neural networks to perform well on such data with fewer $M < N$ hidden layer neurons. Linear procedures such as the Fisher linear discriminant are obviously not expected to give good P_C on such data. As discussed in Appendix B, for large positive values of p' and very small positive values of p, the two-stage MDF approximates the max operation. In this case, $p' = 15$ and $p = 0.25$ were selected automatically in the MDF algorithm; we, thus, expect the MDF to approximate the max function well. This is seen to be the case, as visual inspection of Fig. 8 shows. The P_C performance of the various methods on the test data set are shown in Table 2. The P_C performance of the MDF neural network was excellent ($P_C = 99.7\%$). The SVM with a second-order polynomial kernel also gave good performance ($P_C = 98.0\%$). The SVM contour plot for a second-order polynomial kernel is shown in Fig. 9a. With a larger polynomial kernel order of 19, SVM performance improves slightly to $P_C = 98.6\%$; this is due to the fact that the higher-order polynomial SVM produces a decision boundary that better approximates the max function as seen in Fig. 9b. The P_C performance of the multilayer perceptron neural network (MLP NN) with one neuron in the hidden-layers ($M = 1$), and the Fisher linear discriminant (FLD) are worse with $P_C = 78.4$ and 52.3%, respectively (Table 2), as expected. This is due to the fact that both these methods can yield transforms that curve in only one dimension in the input space. The multilayer perceptron with $M = 2$ hidden-layer neurons has a better $P_C = 97.6\%$ that is comparable to that of the MDF and SVM, since its transformation can curve in both input dimensions. The quadratic (second-order) kernel SVM algorithm chose $V = 15$ support vectors (determined from the training set), resulting in a larger computation time (Table 2).

Our results on 2-D input data show that both the MDF and SVM yield good classification performance on nonlinearly

Table 2
P_C and computation time for the max-distributed 2-D data on the test set using different methods

	MDF NN	MLP NN	MLP NN	FLD	SVM
	$(M, K=1)$	$(M=1)$	$(M=2)$	(1)	$(V=15)$
P_C (%)	99.7	78.4	97.6	52.3	98.0
Comp. time	1	1	2	1	15

separable classes with $S > N$. When only one hidden-layer neuron was used in the MDF and nonlinear PCA neural networks (to simulate high-dimensional input cases), the MDF yielded good P_C while the nonlinear PCA performance was poor since its transformations with $M < N$ cannot curve in all input dimensions.

5.2. Case studies with $S < N$

In this section, we evaluate the performance of the different methods on cases where the number of training samples are less than the input dimensionality, $S < N$. *This is a simulation of high-dimensional input cases, such as image data, with few training examples*. We have seen only one prior experiment of the SVM on cases with $S < N$ (Roobaert & Hulle, 1999). For our case studies, we used $N = 50$ (50-D) input data, with a total of $S = 25$ training samples ($\simeq$12 per class). The number of test samples were 2000 ($\simeq$1000 per class) to provide statistically significant P_C measures. We used the same two hyperspherical and max-class distribution cases as in Section 5.1, but now in a higher (50-D) input space. Note that both of these case studies require full-rank nonlinear transforms for correct class separation (classification). We also consider two other cases, one that involves linearly separable classes, and another where the classes are nonlinearly separable, but decision surfaces with full-rank nonlinear transforms are not needed to correctly classify them.

5.2.1. Case study—1 (linearly separable classes)

We first consider the simplest discrimination case when one class is linearly separable from another class. We consider nonoverlapping, perfectly separable classes with a margin of 0.03 between them, which is an easy classification problem. We expect all the methods to perform comparably on this case study. The P_C performance using the different methods are shown in Table 3. With one hidden-layer neuron, the MDF neural network (NN) P_C performance (99.3%) was similar to that of the MLP NN (99.0%), and a single ODV feature (98.9%). The performance of these last two methods improved slightly when a larger number of hidden-layer neurons M and $K = M$ output features were used. In this case, $p' = 1$ and $p = 1$ were selected automatically in the MDF algorithm, corresponding to a linear transform. The SVM performed the best ($P_C = 99.5\%$). This case study illustrates that on high-dimensional data with $S < N$, all techniques perform similarly when the classes are linearly separable.

5.2.2. Case study—2 (quadratically separable data)

We will consider two quadratically separable cases. We first consider the case when one class encloses another class in only a few ($D = 10$) dimensions in the $S = 50$ dimensional input space. In this case, the optimal discriminant is a quadratic nonlinear transform that curves in only the first $D = 10$ dimensions of the $N = 50$ dimensional input space. For the MDF neural network, we found the P_C training set performance to peak at $p' = 2$ and $p = 1$, i.e. a quadratic transformation was automatically chosen. The performance of the MLP NN and the ODV was good when a large number of hidden-layer neurons (MLP NN) and output features (ODV) were used. The polynomial SVM performed well ($P_C = 96.3\%$) since its transform can curve in a maximum of $S > D$ dimensions, if all S training samples are chosen as support vectors V. This example illustrates that on high-dimensional data with $S < N$, all techniques perform comparably when full-rank nonlinear transformations are not necessary.

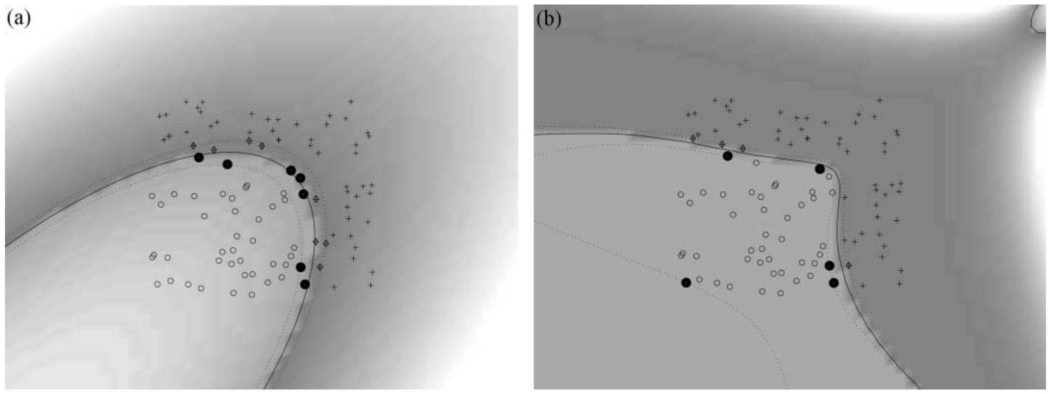

Fig. 9. Plots of 2-D max-distributed training data with the corresponding SVM decision boundaries shown as dark lines and with shaded regions representing SVM output contours for (a) SVM contour plot with $p = 2$ (second-order) : a second-polynomial kernel, and (b) SVM contour plot with $p = 19$: a 19th-order polynomial kernel. The support vectors are the filled-in circles close to the decision boundaries.

Table 3
P_C values on 50-D linearly-separable test set data using different methods with a total of $S = 25$ training samples

	MDF NN	MLP NN		ODV		SVM
	$M = 1$	$M = 1$	$M = 5$	1	5	$(V = 7)$
P_C (%)	99.3	99.0	99.4	98.9	99.2	99.5
Comp. time ($\times \mathcal{O}(N)$)	1	1	5	1	5	7

We then considered a case where one class *completely* encloses the other class, a closed decision boundary is necessary; therefore, a full-rank nonlinear (quadratic) transformation can separate the class best. Since $S < N$, we do not expect the multilayer perceptron neural network or SVMs to yield good P_C performance, as discussed earlier in Section 4. The MDF neural network can produce closed-decision surfaces, even when $S < N$, or $S \ll N$ as we proved in Section 3 and, thus, we expect its P_C to be good for this case study.

For the MDF neural network, we found the P_C training set performance to peak at $p' = 2$ and $p = 1$. With these MDF neural network parameters, we obtained excellent P_C performance on the 2000 test samples with the same class distributions. With only $M = K = 1$ neurons in the MDF, $P_C = 89.4\%$ was obtained (Table 4). Increasing M and K did not improve P_C for the MDF. Even though the MDF transform is a quadratic one (which is the optimal transform for this case), P_C performance is not perfect due to the small training set size which leads to inaccuracies in estimation of the MDF neural network weights and the thresholds to use. In contrast, the multilayer perceptron neural network (MLP NN) yielded only $P_C = 54.4\%$ with $M = 1$ hidden-layer neuron or a better $P_C \simeq 75\%$ performance with $M = 41$ neurons; this occurs since its transforms now curve in more dimensions (at most 41 dimensions) in the input space. However, its performance is still worse than that of the MDF neural network. The linear ODV discriminant (the extension of the Fisher linear discriminant to more than one discriminating feature) gave comparable P_C performance to that of the multilayer perceptron. The SVM classifier yielded only $P_C = 52.6\%$ with $V = 13$ support vectors for a second-order polynomial kernel. This low P_C occurs because the polynomial kernel SVM can only produce nonlinear transforms that curve in a maximum of $V = 13$ (support vector set size) dimensions in the input space, as we proved earlier in Section 4.3. A similar degradation in SVM performance with high-dimensional inputs has been noted (Roobaert & Hulle, 1999) on other data when $N = 32 \times 32 = 1024$ input image pixel values were used with only $S = 4$ training views per object. On this other data, SVM P_C performance was seen to improve when only average RGB color information from each pixel was used ($N = 3$) as inputs to the SVM classifier rather than pixel data; thus, SVM performance was found to be significantly better when $S > N$ compared with $S \ll N$. In our case study, *the SVM P_C performance is worse than that of the multilayer perceptron because the nonlinear transforms in the SVM are constrained by the support vector set size ($V < S = 25$ in this case), while those in the multilayer perceptron neural network are limited by the number of second-layer neurons M used*; we used $M > S$. The number of weights in the neural network should be less than the training set size for good generalization (Baum & Haussler, 1989). This is not satisfied here and generalization problems were noticed in our tests. When a larger number of hidden-layer neurons were used, test-set P_C performance increased slightly to $P_C = 78.2\%$ with $M = 55$ hidden-layer neurons, did not change significantly with larger $M = 60$, $M = 65$, and decreased with larger M.

5.2.3. Case study—3 (max distribution)

We used a max distribution of dimension $N = 50$ for our last case study with a total training set size of 25 samples ($\simeq$12 per class) and test set size of 2000 ($\simeq$1000 per class). A gap of 0.03 is kept between the two classes for both training and test set data. Table 5 gives P_C results of the various methods on this dataset. From training data samples, polynomial values of $p' = 9$ and $p = 0.2$ were automatically selected for the MDF neural network. With these parameter values, we obtained $P_C = 76.2\%$ with $M = K = 1$ neurons and the best $P_C = 77.3\%$ with $M = K = 11$ MDF neurons. In contrast, the multilayer perceptron neural network (MLP NN) gave only $P_C = 60.3\%$ with $M = 41$ neurons; a slightly better $P_C = 65.4\%$ performance was obtained with $M = 55 > S$ neurons. The SVM gave $P_C = 50.9\%$ with $V = 24$ support vectors. As before, the SVM yielded worse P_C results since the rank of its transformation is limited by the training set size, while the transformations for the multilayer perceptron neural network are limited by the number of hidden-layer neurons. The ODF performance was comparable to that of the multilayer perceptron neural network. Therefore, the MDF neural network yields the best

Table 4
P_C values on 50-D hyperspherically-distributed test set data using different methods with a total of only $S = 25$ training samples

	MDF NN		MLP NN		ODV		SVM
	$M = 1$	$M = 5$	$M = 1$	$M = 41$	1	41	$(V = 13)$
P_C (%)	89.4	87.2	54.4	74.7	44.6	73.1	52.6
Comp. time ($\times \mathcal{O}(N)$)	1	5	1	41	1	41	13

Table 5
P_C values on 50-D max-distributed test set data using different methods with a total of $S = 25$ training samples

	MDF NN		MLP NN		ODV		SVM
	$M = 1$	$M = 11$	$M = 1$	$M = 41$	1	41	($V = 24$)
P_C (%)	76.2	77.3	46.9	60.3	51.7	57.6	50.9
Comp. time ($\times \mathcal{O}(N)$)	1	11	1	41	1	41	24

P_C results and the least on-line computation time (bottom row in Table 5).

6. Summary

We have presented a new nonlinear feature extraction procedure in a neural network formulation for data classification. The weights of the neural network are obtained in closed-form, thereby overcoming limitations with iterative techniques. The activation functions for the input and hidden-layer neurons are adapted. We proved theoretical properties on the types of nonlinear transforms produced by this neural network. It is shown to yield more general transforms than other well-known nonlinear neural networks and kernel-based methods. Results on synthetic data are shown to support these theoretical analyses. We have shown that when $S < N$ and *full-rank nonlinear transformations are needed* for discrimination, the MDF neural network outperforms the polynomial SVM and multilayer perceptron neural networks; when nonlinear transforms that curve in only a few input dimensions are necessary for discrimination, all methods perform comparably.

Our initial results with the MDF are encouraging and indicative of its use in high-dimensional cases. Our prior work on real (image) data sets in active vision and product inspection applications (Talukder & Casasent, 1998) has also shown its advantages in real problems. Future work could address further comprehensive tests on standard image data sets.

Acknowledgements

This work was supported by the Cooperative State Research, Education, and Extension Service, US Department of Agriculture, under agreement nos. 97-35503-4532 and 98-35503-6340. Any opinions, findings, conclusions, or recommendations expressed in this publication are those of the authors and do not necessarily reflect the view of the US Department of Agriculture.

Appendices. Proofs

In these appendices, we prove two theorems associated with the nonlinear transformations produced by the nonlinear MDF, nonlinear PCA neural networks, and kernel based methods.

Appendix A. Transformations produced by a nonlinear mapping of a linear combination of input data

Theorem 1. Any nonlinear transformation $\Psi(\underline{x})$ of an input datum $\underline{x}$ involving a linear weighted combination of the input mapped through a nonlinear function $f(\ )$ of the form $\Psi(\underline{x}) = f(\underline{w}^T\underline{x})$ folds or curves in only one dimension in the input space with a principal orientation along $\underline{w}$.

Therefore, a nonlinear transform of the form $\Psi(\underline{x}) = \sum_{m=1}^{M} a_m f(\underline{w}_m^T\underline{x})$ folds or curves in a maximum of M dimensions in the input space. The principal orientation of this transform lies in the subspace spanned by the weight vectors $\underline{w}_1, \underline{w}_2, \ldots, \underline{w}_M$.

Proof. We derive our proof for this theorem using Taylor series expansion to describe the nonlinear transforms. We make use of two assumptions that appear to be intuitive enough and therefore will not be proven. We shall make use of the fact that the *product* of two nonlinear transforms, each of which curves in a subspace in the input space, itself curves in the same subspace. Similarly, the *sum* of two nonlinear transforms, each of which curves in a subspace in the input space, itself curves in the same subspace.

We first express the nonlinear transformations that involve a linear weighted combination of the input $\underline{x}$ passed through a nonlinear function $f(\ )$, $\Psi(\underline{x}) = f(\underline{w}^T\underline{x})$, as a Taylor series expansion of a single variable x (Spiegel, 1983):

$$f(\underline{w}^T\underline{x}) = a_0 + a_1(\underline{w}^T\underline{x}) + a_2(\underline{w}^T\underline{x})^2 + a_3(\underline{w}^T\underline{x})^3 + \cdots.$$

The coefficients, $a_0, a_1, \ldots$ are determined by the n-th order derivatives of the function $f(y)$ at $y = 0$. We consider vector notation and write the Taylor series as

$$f(\underline{w}^T\underline{x}) = a_0 + a_1(\underline{w}^T\underline{x}) + a_2(\underline{x}^T\underline{w}\underline{w}^T\underline{x}) + a_3(\underline{x}^T\underline{w}\underline{w}^T\underline{x})(\underline{w}^T\underline{x}) + \cdots. \quad \text{(A1)}$$

If we write each quadratic term $\underline{x}^T\underline{w}\underline{w}^T\underline{x}$, as a matrix transformation, $\underline{x}^T\underline{A}_w\underline{x}$, where $\underline{A}_w = \underline{w}\underline{w}^T$, the nonlinear transform can be written in terms of the matrix $\underline{A}_w$ as

$$f(\underline{w}^T\underline{x}) = a_0 + a_1(\underline{w}^T\underline{x}) + a_2(\underline{x}^T\underline{A}_w\underline{x}) + a_3(\underline{x}^T\underline{A}_w\underline{x}) \times (\underline{w}^T\underline{x}) + \cdots + a_3(\underline{x}^T\underline{A}_w\underline{x})(\underline{x}^T\underline{A}_w\underline{x})\cdots. \quad \text{(A2)}$$

We ignore the coefficients a_0, a_1, etc. for now without any loss in generality and discuss the effect of each term in Eq. (A2).

We shall show that each term in Eq. (A2) curves along the same weight vector $\underline{w}$ and, therefore the sum of these terms also curves along $\underline{w}$.

Term 1 in Eq. (A2) is a constant, term 2 in Eq. (A2) is a linear mapping along the vector $\underline{w}$. The third term in Eq. (A2) is a quadratic term of the form $\underline{x}^T\underline{A}_w\underline{x}$ where $\underline{A}_w = \underline{w}\underline{w}^T$ is of rank one. It has been shown in prior work (Weber & Casasent, 1998) that such a quadratic transform curves only in one dimension. From linear algebra, the pricipa1 orientation of this quadratic transform occurs along the vector $\underline{w}$, since the principal eigenvector is parallel to $\underline{w}$.

The fourth term is a product of a quadratic term $\underline{x}^T\underline{A}_w\underline{x}$ and a linear term $\underline{w}^T\underline{x}$. The quadratic term has a principal orientation along $\underline{w}$ as we discussed earlier, and curves or folds in one dimension along $\underline{w}$. The linear term also changes linearly along the vector $\underline{w}$. Therefore, the product of these terms, that results in a third-order polynomial term also curves or folds along the vector $\underline{w}$.

The fifth term is a fourth order polynomial that is expressed as a product of two quadratics, $(\underline{x}^T\underline{A}_w\underline{x})(\underline{x}^T\underline{A}_w\underline{x})$. Therefore, since each of the quadratic terms folds along $\underline{w}$, the resultant fourth-order polynomial formed by the product also curves or folds along $\underline{w}$.

The same argument holds for *all* higher-order polynomial terms in the Taylor series expansion for $f(\underline{w}^T\underline{x})$. Therefore, the resultant sum of all polynomial orders in Eq. (A2), each of which folds along $\underline{w}$, itself folds only in one dimension in the input space along the weight vector $\underline{w}$. This proves the first part of this theorem that *any* nonlinear mapping of the form $\Psi(\underline{x}) = f(\underline{w}^T\underline{x})$ curves in only one dimension in the input space; the curvature occurs along $\underline{w}$.

We will now prove that a linear combination of M such nonlinear transforms curve or fold in a maximum of M dimensions in the input space. A transform of the form

$$\Psi(\underline{x}) = \sum_{m=1}^{M} a_m f(\underline{w}_m^T\underline{x}) \tag{A3}$$

involves M vector inner products of the input data with M weight vectors $\underline{w}_1, \underline{w}_2, \ldots, \underline{w}_M$, each passed through a nonlinear mapping $f(\ )$. Each term in the summation in Eq. (A3) curves in one dimension in the input space as we showed earlier, with the corresponding principal orientation along $\underline{w}_m$. Therefore, the sum of M such nonlinear transforms as in Eq. (A3) *curves in a maximum of M dimensions* in the input space with the *principal orientation in the subspace spanned by the weight vectors* $\underline{w}_1, \underline{w}_2, \ldots, \underline{w}_M$. Thus, the maximum rank of this transform is M.

Appendix B. Nonlinear MDFs as morphological operators

Theorem 2. The two-stage nonlinear MDF for image input data specializes to the p-norm operation under certain conditions. It also emulates the min (morphological erosion) operation on image data and the max (morphological dilation) operation on image data for a specific case.

Proof. We derive our proof for this theorem in two steps. We first prove that for specific values for the nonlinear MDF coefficients and p' and p, the nonlinear MDF is exactly the same as a p-norm transform. We then prove that the nonlinear MDF can perform morphological or linear filtering operations.

The p-norm of a vector $\underline{x}$ is $z_p = (x_1^p + x_2^p + x_3^p + \cdots + x_n^p)^{1/p}$, where the value of p is real, and can vary from $-\infty$ to $+\infty$. The weighted p-norm of a vector $\underline{x}$ is (Gader, 1992)

$$z_p = (\phi_1 x_1^p + \phi_2 x_2^p + \phi_3 x_3^p + \cdots + \phi_n x_n^p)^{1/p}, \tag{B1}$$

where $p \in \{-\infty, +\infty\}$.

To derive the fact that the nonlinear MDF specializes to the p-norm operation, we note that *our two-step transform can be written as a single-step nonlinear transform of the input image* $\underline{x}$, $\underline{y}_K = \underline{\Phi}_K^T(\underline{\Phi}_M'^T\underline{x}.^{p'}).^p$, where.^{p} denotes each element in the vector raised to the power p. If $\underline{\Phi}_K$ is a diagonal identity matrix (with $K = M$), then the output features are $\underline{y}_K = (\underline{\Phi}_M'^T\underline{x}.^{p'}).^p$. If $p = 1/p'$, then $\underline{y}_K = (\underline{\Phi}_M'^T\underline{x}.^{p'}).^{1/p'}$. Writing the vector $\underline{\phi}'_m$ as $\underline{\phi}'_m = [\,\phi'_{1m} \quad \phi'_{2m} \quad \cdots \quad \phi'_{Nm}\,]^T$, the m-th output MDF feature is

$$y_m = (\phi_{1m} x_1^p + \phi_{2m} x_2^p + \cdots + \phi_{Nm} x_N^p)^{1/p'}. \tag{B2}$$

This corresponds to a set of M (or K) weighted p-norm transformations as in Eq. (B1).

To prove the morphological properties of the nonlinear MDF transform, we consider the case when $p' \to +\infty$ *and* $p = 1/p'$, with the transformation matrix $\underline{\Phi}_K$ in the second stage of the nonlinear MDF being an identity matrix with $K = M$, as before. Then, the m-th output MDF feature is given by Eq. (B2). We will now prove that the nonlinear MDF transform can emulate the morphological dilation or the max operation. Without any loss in generalization, we assume that among the N pixels in the input image $\underline{x}$, x_1 (pixel 1) has the largest (maximum) gray value. Then, Eq. (B2) can be written as

$$y_m = [x_1^{p'}(\phi'_{1m} + \phi'_{2m}(x_2/x_1)^{p'} + \cdots + \phi'_{Nm}(x_N/x_1)^{p'})]^{1/p'}. \tag{B3}$$

If all the MDF transform coefficients are equal, i.e. $\phi'_{1m} = \phi'_{2m} = \cdots = \phi'_{Nm}$, the output feature simplifies to

$$y_m[x_1^{p'}(1 + (x_2/x_1)^{p'} + \cdots + (x_N/x_1)^{p'})]^{1/p'}, \tag{B4}$$

with a scaling factor ϕ'_{1m} which may be ignored. Note that all of the ratio terms in Eq. (B4) (x_2/x_1), (x_3/x_1), …, (x_N/x_1) are *less than* 1, since pixel x_1 is assumed to have the largest gray value, $(x_2/x_1) < 1$, $(x_3/x_1) < 1$, etc. If $p' \to +\infty$, then all of the ratio terms $\to 0$ in the limit. Therefore, the output feature

$$y_m = [x_1^{p'}(1 + 0 + \cdots + 0)]^{1/p'} = x_1, \tag{B5}$$

which is the *maximum* of all the pixels in the image $\underline{x}$. This is exactly the same as the mathematical morphology dilation operation at the image center, where the structuring element or template is the same size as the input image.

The proof that the nonlinear MDF can emulate the morphological erosion or the min operation is very similar to the proof above. Assuming that among the N pixels in the input image $\underline{x}$, x_1 (pixel 1) has the lowest (minimum) gray value, and $p' \rightarrow -\infty$ *and* $p = 1/p'$, the output feature in Eq. (B2) can be written as

$$y_m = [x_1^{p'}(\phi'_{1m} + \phi'_{2m}(x_2/x_1)^{p'} + \cdots + \phi'_{Nm}(x_N/x_1)^{p'})]^{1/p'}. \tag{B6}$$

All of the ratio terms in Eq. (B6) (x_2/x_1), (x_3/x_1), …, (x_N/x_1) are *greater than* 1 since pixel x_1 is assumed to have the smallest gray value; $(x_2/x_1) > 1$, etc. If $p' \rightarrow -\infty$, then all of the ratio terms in Eq. (B6) $\rightarrow 0$ in the limit, and the output feature is $y_m = x_1$, which is the minimum of all the pixels in the image $\underline{x}$. This is exactly the same as the morphological erosion operation at the image center, where the structuring element or template is the same size as the input image.

References

Baldi, P., & Hornik, K. (1989). Neural networks and principal component analysis: learning from examples without local minima. *Neural Networks*, *2* (1), 53–58.

Barlow, H. B. (1961). Possible principles underlying the transmission of sensory messages. In W. A. Rosenbluth, *Sensory communication* (pp. 217–234). Cambridge, MA: MIT Press.

Baum, E. B., & Haussler, D. (1989). What size net gives valid generalization? *Neural Computation*, *1* (1), 151–160.

Bishop, C. (1995). *Neural networks for pattern recognition*, Oxford: Oxford University Press.

Boser, B., Guyon, I., & Vapnik, V. (1992). A training algorithm for optimal margin classifiers. In *Proc. Fifth Annual Workshop of Computational Intelligence* (Vol. 5, pp. 144–152).

Burges, C. J. C. (1998). A tutorial on support-vector machines for pattern recognition. *Data Mining and Knowledge Discovery*, *2* (2), 121–167.

Casasent, D., & Natarajan, S. (1995). A classifier neural net with complex-valued weights and square-law nonlinearities. *Neural Networks*, *8* (6), 989–998.

Casasent, D., Neiberg, L., & Sipe, M. (1998). Feature space trajectory distorted object representation for classification and pose estimation. *Optical Engineering*, *37* (3), 914–923.

Cortes, C., & Vapnik, V. (1995). Support-vector networks. *Machine Learning*, *20*, 1–25.

Cover, T. M. (1965). Geometrical and statistical properties of systems of linear inequalities with applications in pattern recognition. *IEEE Trans. Electronic Computers*, *EC-14*, 326–334.

Denoeux, T., & Lengelle, R. (1993). Initializing backpropagation networks with prototypes. *Neural Networks*, *6*, 351–363.

Dougherty, E. (1986). An introduction to morphological image processing. In. D. C. O'Shea (Ed.), *Tutorial texts in optical engineering SPIE* (Vol. TT9). Bellingham, WA.

Duda, R. O., & Hart, P. E. (1973). *Pattern classification and scene analysis*, New York: John Wiley.

Fisher, R. A. (1950). *Contributions to mathematical statistics*, New York: John Wiley.

Foley, D. H., & Sammon, J. W. (1975). An optimal set of discriminant vectors. *IEEE Trans. Comput.*, *C-24*, 281–289.

Gader, P. (1992). Template generation for pattern classification. *Proc. SPIE: Image Algebra and Morphological Image Processing III*, *1769*, 72–81.

Gheen, G. (1994). Distortion invariant Volterra filters. *Pattern Recognition*, *27* (4), 569–576.

Gunn, S. R. (1997). Support vector machines for classification and regression. *Technical Report, Image Speech and Intelligent Systems Research Group*, *1* (1), 1–52.

Hamamoto, Y., Kanaoka, T., & Tomita, S. (1993). On a theoretical comparison between the orthonormal discriminant vector method and discriminant analysis. *Pattern Recognition*, *26* (12), 1863–1867.

Heywood, M., & Noakes, P. (1995). A framework for improved training of Sigma–Pi networks. *IEEE Trans. Neural Networks*, *6* (4), 893–903.

Karhunen, J., & Joutsensalo, J. (1994). Representation and separation of signals using nonlinear PCA type learning. *Neural Networks*, *7* (1), 113–127.

Karhunen, J., & Joutsensalo, J. (1995). Generalizations of principal component analysis, optimization problems, and neural networks. *Neural Networks*, *8* (4), 549–562.

Kramer, M. A. (1991). Nonlinear principal component analysis using autoassociative neural networks. *AIChE Journal*, *37* (2), 233–243.

Lippmann, R. P. (1987). An introduction to computing with neural nets. *IEEE ASSP Mag.*, *4* (2), 4–22.

Malthouse, E. C. (1998). Limitations of nonlinear PCA as performed with generic neural networks. *IEEE Trans. Neural Networks*, *9* (1), 165–173.

Osuna, E. E., Freund, J. R., & Girosi, F. (1997). *Support-vector machines: training and applications*. Technical Report, A.I. Memo No. 1602 and C.B.C.L. Paper No. 144, Massachusetts Institute of Technology.

Roobaert, D., & Hulle, M. V. (1999). View-based 3D object recognition with support vector machines. In *Proceedings IEEE International Workshop on Neural Networks for Signal Processing* (Vol. 1).

Ruck, D., Rogers, S., Kabrisky, M., Oxley, M., & Suter, B. (1990). The multilayer perceptron as an approximation to a Bayes optimal discriminant function. *IEEE Trans. Neural Networks*, *1* (4), 296–298.

Scholkopf, B., Burges, C., & Vapnik, V. (1995). Extracting support data for a given task. In U. M. Fayyad & R. Uthurusamy (Eds.), *Proceedings First International Conference on Knowledge Discovery and Data Mining* (pp. 252-257).

Scholkopf, B., Smola, A., & Muller, K. (1998). Nonlinear component analysis as a kernel eigenvalue problem. *Neural Computation*, *10*, 1299–1319.

Softky, W. S., & Kammen, D. M. (1991). Correlations in high-dimensional or asymmetric data sets: Hebbian neuronal processing. *Neural Networks*, *4* (3), 337–348.

Spiegel, M. R. (1983). *Schaums outline series: theory and problems of advanced mathematics for scientists and engineers*, New York: McGraw-Hill.

Strang, G. (1980). *Linear algebra and its applications*, Orlando: Harcourt Brace Jovanovich.

Talukder, A., & Casasent, D. (1998). General methodology for simultaneous representation and discrimination of multiple object classes. *Optical Engineering, Special Issue on Advanced Recognition Techniques*, *37* (3), 904–913.

Talukder, A., & Casasent, D. (1999). Pose invariant recognition of faces at unknown aspect views (invited paper). *International Joint Conference on Neural Networks*, *5*, 3286–3290.

Telfer, B., & Casasent, D. (1993). Minimum-cost associative processor for piecewise-hyperspherial classification. *Neural Networks*, *6* (8), 1117–1130.

Vapnik, V. N. (1995). *The nature of statistical learning theory*, New York: Springer.

Weber, D., & Casasent, D. (1998). The extended piecewise quadratic neural network. *Neural Networks*, *11*, 837–850.